Customizing Istio Metrics

4 minute read   page test

This task shows you how to customize the metrics that Istio generates.

Istio generates telemetry that various dashboards consume to help you visualize your mesh. For example, dashboards that support Istio include:

- Grafana
- Kiali
- Prometheus

By default, Istio defines and generates a set of standard metrics (e.g. `requests_total`), but you can also customize them and create new metrics.

Custom statistics configuration

Istio uses the Envoy proxy to generate metrics and provides its configuration in the `EnvoyFilter` at manifests/charts/istio-control/istio-discovery/templates/telemetryv2_1.11.yaml.

Configuring custom statistics involves two sections of the `EnvoyFilter`: `definitions` and `metrics`. The `definitions` section supports creating new metrics by name, the expected value expression, and the metric type (`counter`, `gauge`, and `histogram`). The `metrics` section provides values for the metric dimensions as expressions, and allows you to remove or override the existing metric dimensions. You can modify the standard metric definitions using `tags_to_remove` or by re-defining a dimension. These configuration settings are also exposed as istioctl installation options, which allow you to customize different metrics for gateways and sidecars as well as for the inbound or outbound direction.

For more information, see Stats Config reference.

Before you begin

Install Istio in your cluster and deploy an application.

Alternatively, you can set up custom statistics as part of the Istio installation.

The Bookinfo sample application is used as the example application throughout this task.

Enable custom metrics

1. The default telemetry v2 `EnvoyFilter` configuration is equivalent to the following installation options:

```yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    telemetry:
      v2:
        prometheus:
          configOverride:
            inboundSidecar:
              disable_host_header_fallback: false
            outboundSidecar:
              disable_host_header_fallback: false
            gateway:
              disable_host_header_fallback: true
```

To customize telemetry v2 metrics, for example, to add `request_host` and `destination_port` dimensions to the `requests_total` metric emitted by both gateways and sidecars in the inbound and outbound direction, change the installation options as follows:

You only need to specify the configuration for the settings that you want to customize. For example, to only customize the sidecar inbound `requests_count` metric, you can omit the `outboundSidecar` and `gateway` sections in the configuration. Unspecified settings will retain the default configuration, equivalent to the explicit settings shown above.

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    telemetry:
      v2:
        prometheus:
          configOverride:
            inboundSidecar:
              metrics:
                - name: requests_total
                  dimensions:
                    destination_port: string(destination.po
rt)
                    request_host: request.host
            outboundSidecar:
              metrics:
                - name: requests_total
                  dimensions:
                    destination_port: string(destination.po
rt)
                    request_host: request.host
            gateway:
              metrics:
                - name: requests_total
                  dimensions:
                    destination_port: string(destination.po
rt)
                    request_host: request.host
```

2. Apply the following annotation to all injected pods with the list of the dimensions to extract into a Prometheus time series using the following command:

This step is needed only if your dimensions are not already in DefaultStatTags list

```
apiVersion: apps/v1
kind: Deployment
spec:
  template: # pod template
    metadata:
      annotations:
        sidecar.istio.io/extraStatTags: destination_port,re
quest_host
```

To enable extra tags mesh wide, you can add `extraStatTags` to your mesh config:

```
meshConfig:
  defaultConfig:
    extraStatTags:
     - destination_port
     - request_host
```

Verify the results

Send traffic to the mesh. For the Bookinfo sample, visit `http://$GATEWAY_URL/productpage` in your web browser or issue the following command:

```
$ curl "http://$GATEWAY_URL/productpage"
```

`$GATEWAY_URL` is the value set in the Bookinfo example.

Use the following command to verify that Istio generates the data for your new or modified dimensions:

```
$ kubectl exec "$(kubectl get pod -l app=productpage -o jsonpath
='{.items[0].metadata.name}')" -c istio-proxy -- curl -sS 'local
host:15000/stats/prometheus' | grep istio_requests_total
```

For example, in the output, locate the metric `istio_requests_total` and verify it contains your new dimension.

It might take a short period of time for the proxies to start applying the config. If the metric is not received, you may retry sending requests after a short wait, and look for the metric again.

Use expressions for values

The values in the metric configuration are common expressions, which means you must double-quote strings in JSON, e.g. "'string value'". Unlike Mixer expression language, there is no support for the pipe (`|`) operator, but you can emulate it with the `has` or `in` operator, for example:

```
has(request.host) ? request.host : "unknown"
```

For more information, see Common Expression Language.

Istio exposes all standard Envoy attributes. Peer metadata is available as attributes `upstream_peer` for outbound and `downstream_peer` for inbound with the following fields:

Field	Type	Value
`name`	`string`	Name of the pod.
`namespace`	`string`	Namespace that the pod runs in.
`labels`	`map`	Workload labels.

owner	string	Workload owner.
workload_name	string	Workload name.
platform_metadata	map	Platform metadata with prefixed keys.
istio_version	string	Version identifier for the proxy.
mesh_id	string	Unique identifier for the mesh.
app_cont	list<s	List of short names for

ainers	tring>	application containers.
cluster_id	string	Identifier for the cluster to which this workload belongs.

For example, the expression for the peer `app` label to be used in an outbound configuration is `upstream_peer.labels['app'].value`.

For more information, see configuration reference.